RL-NP-93-2
Final Technical Report
August 1993

SPEAKEASY SOFTWARE DEVELOPMENT

The Analytic Sciences Corporation (TASC)

Ms. Patricia J. Baskinger, Larry Ozarow, and Ms. Mary Carol Chruscicki

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

Unedited version of this report, RL-NP-93-2 dated August 1993, is being sent to the Defense Technical Information Center (DTIC) for archiving and subsequent referral through the DTIC Technical Report Data Base.

RL-NP- 93-2 has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

APPROVED: William J. Maxey

WILLIAM J. MAXEY
Project Engineer

FOR THE COMMANDER: John A. Graniero

JOHN A. GRANIERO
Technical Director
Command, Control & Communications Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1993	Final May 91 - Apr 93

4. TITLE AND SUBTITLE
SPEAKEASY SOFTWARE DEVELOPMENT

5. FUNDING NUMBERS
C - F30602-89-D-0050, Task 0005
PE - 63737D
PR - 4100
TA - Q5
WU - H8

6. AUTHOR(S)
Ms. Patricia J. Baskinger, Larry Ozarow, and
Ms. Mary Carol Chruscicki

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Analytic Sciences Corporation (TASC)
555 French Road
New Hartford NY 13413-0895

8. PERFORMING ORGANIZATION REPORT NUMBER
STR-6005-2

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (C3BB)
525 Brooks Road
Griffiss AFB NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-NP-93-2

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: W. John Maxey/C3BB/(315) 330-3617

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Speakeasy Software Development Project had three primary objectives. The first objective was to perform Independent Verification and Validation (IV & V) of the software and documentation associated with the signal processor being developed by Hazeltine and TRW under the Speakeasy program. The IV & V task also included an analysis and assessment of the ability of the signal processor software to provide LPI communications functions. The second objective was to assist in the enhancement and modification of an existing Rome Lab signal processor workstation. Finally, TASC developed project management support tools and provided program management support to the Speakeasy Program Office.

14. SUBJECT TERMS
Software Development Management, LPI Communications

15. NUMBER OF PAGES
56

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

The Speakeasy Software Development Project had three primary objectives. The first objective was to perform Independent Verification and Validation (IV&V) of the software and documentation associated with the signal processor being developed by Hazeltine and TRW under the Speakeasy program. The IV&V task also included an analysis and assessment of the ability of the signal processor software to provide LPI communications functions. The second objective was to assist in the enhancement and modification of an existing Rome Lab signal processor workstation. Finally, TASC developed project management support tools and provided program management support to the Speakeasy Program office.

For the IV&V task (Chapter 3), we first focused on reviewing the system and software documentation and requirements traceability. We then reviewed the software development processes being employed by the Speakeasy development team. Finally, we evaluated the advanced software-related technologies being developed under the sponsorship of the Balanced Technology Initiative (BTI).

As part of our efforts to enhance the Rome Laboratory (RL) Signal Processing Workstation (Chapter 4), we first delivered our previously developed Vulnerability Metric software model. We then extended this model with the development of the Digital Density Detector module. In addition, we did an analysis of the error probability of the variable rate waveform computed under two different demodulation regimes: bit-by-bit decoding and symbol-by-symbol decoding.

Our program management support (Chapter 5) focused on an initial survey of the major software development risks and initiation of efforts towards resolving these risks. Our efforts found significant deficiencies in the Speakeasy requirements (system and software), documentation, and software development process. We identified six high risk areas that required immediate attention and became our focus throughout this effort. Several software development management techniques coupled with proactive management involvement were initiated to facilitate the successful completion of this large software development effort. We also provided program management support including cost/schedule system evaluation, system document preparation and configuration management, and the implementation of an automated action item tracking system.

1. INTRODUCTION

The Speakeasy Software Development Project had three primary objectives. The first objective was to perform Independent Verification and Validation (IV&V) of the software and documentation associated with the signal processor being developed by Hazeltine and TRW under the Speakeasy program. The IV&V task also included an analysis and assessment of the ability of the signal processor software to provide LPI communications functions. The second objective was to assist in the enhancement and modification of an existing Rome Lab signal processor workstation. Finally, TASC developed project management support tools and provided program management support to the Speakeasy Program office.

Three tasks were defined to meet the objectives of this effort. Under Task I, we applied IV&V techniques covering participation in design reviews, analysis of the software development process for Speakeasy, and evaluation of related Speakeasy documentation and code. Under Task II, we provided assistance in modifying and enhancing an existing in-house workstation that supports the design and evaluation of low probability of intercept (LPI) communications and Advanced (LPI) Waveform analysis support. Under Task III, we integrated the tools and capabilities needed to provide program management support functions for the Speakeasy Program Office.

This report covers a twenty four month period of the Speakeasy software development effort, from 3 May 1991 to 30 April 1993. This period of time covered most of the Detailed Design Phase and approximately half of the Coding Phase of the Speakeasy program, as defined by DoD-STD-2167A. During this time, the focus and funding of the Speakeasy program expanded from developing only the advanced programmable signal processor for a next generation radio, to developing the other needed components for an advanced radio system. Consequently, much more emphasis was placed on system engineering and integration, the software development process, maintainability, portability, and security related issues. This was reflected in the change in program name from the Tactical Anti-Jam Programmable Signal Processor to Speakeasy, and an increased emphasis on the quality of the software being developed.

The following chapter provides the reader with more background on the Speakeasy program. Chapter 3 highlights our IV&V efforts under Task I. Task II, LPI analysis and enhancement of the RL Signal Processing Workstation, is discussed in Chapter 4. And finally, the key points of our program management support and lessons learned are summarized in Chapter 5.

2. BACKGROUND

The Speakeasy program is a joint service program to develop a tactical multimode, multiband radio to satisfy the future communications requirements of the DoD and other government agencies. The objective of the Speakeasy program is to develop a programmable, reconfigurable, and modular radio system using advanced technology to support 21st century communication requirements. Speakeasy is a state-of-the-art tactical radio system that has the capability of emulating many different, current and future, fielded radios. The program will provide the radio resources to effectively operate in highly mobile, hostile, and Electronic Countermeasure intensive environments. The radio system will provide the survivability, reliability, interoperability, and adaptability required by physically dispersed and functionally distributed command and control structures.

Speakeasy will incorporate the latest technologies available in signal processing, frequency synthesis, filters, antenna designs, and many other areas which are currently being researched. Hazeltine Corporation (the prime contractor) and TRW (the major subcontractor) form the core of the Phase I Speakeasy development team. Hazeltine's expertise includes in-depth understanding of Anti-Jam/Low Probability of Intercept design issues and radio hardware development. Through the Integrated Communications Navigation Identification Avionics (ICNIA) program, TRW provides a strong awareness of the problems inherent to programmable, concurrent signal processing.

The Speakeasy software is being developed in Ada, with a small amount of assembly code being used for machine dependent and time critical functions. The Speakeasy Statement of Work (SOW) defines a tailored DoD-STD-2167A software development. The basic products of the Phase I program are an Advanced Development Model (ADM) which is a functional prototype focusing on the signal and data processing aspects of the multimode, multiband radio, and documentation (reports and specifications) to guide the Phase II (post-1994) effort.

Some of the overall design requirements for the radio include modularity, expandability, reliability and utilizing Pre-Planned Product Improvement (P3I) concepts. A major technical challenge of this program is the development of real time mission critical software written in Ada, which will evolve as future requirements are levied upon the program. The software must be flexible and portable to the degree that the system can easily accommodate new hardware advances. More importantly, the software must have a high degree of maintainability and reliability so that it can be transitioned efficiently into an operational/fieldable system.

This requirement to create an expandable, modular and reconfigurable system has forced the design engineers to utilize the flexible nature of software to provide a more integrated package of functionality in what has conventionally been an exclusively hardware-based system. Consequently, a major goal of this effort was to provide Speakeasy's software development managers with insight into methods of managing large software efforts through the use of proven software engineering concepts and problem prevention techniques.

3. IV&V TECHNICAL TASKS

In this chapter, we describe our efforts under Task I. Our IV&V technical support first focused on reviewing the efforts to establish suitable system requirements baselines. A major goal of the Speakeasy program office was that the software developed in Phase I form the foundation of and be largely reused for the Phase II Speakeasy development. This goal took us from looking at the overall system requirements to establishing a very detailed software requirements baseline. As the development got underway, we then began reviewing the contractors' software development process.

During this period, we generated and delivered eleven Technical Information Reports (TIRs) (see Table 1.1-1) to the government covering the various aspects of our IV&V analysis and evaluation of the Speakeasy design, schedule, deliverables, and software development process. This report presents a high level review from the perspective of the three subtasks. More detailed information can be obtained by requesting the TIRs from the Speakeasy Program Office.

Table 1.1-1 Speakeasy Software Development Technical Information Reports

NUMBER	REPORT TITLE
TIR6005-1	Preliminary Risk Management Assessment
TIR6005-2	Evaluation of Compilers For The Advanced Signal Processor
TIR6005-3	Review of Top Level Speakeasy Documents and Requirements Traceability
TIR6005-4	Evaluation of Initial Hazeltine Software Development Products
TIR6005-5	Evaluation of Compilers For The Baseline Microprocessor
TIR6005-6	Hazeltine Program Management Evaluation And Action Plan
TIR6005-7	Cost Estimation For The Speakeasy Software Development
TIR6005-8	Evaluation of Speakeasy Software Documentation
TIR6005-9	Review of the Executive Code
TIR6005-10	A Comparison of NSA DS80 To DODSTD2167A
TIR6005-11	Review of Speakeasy Software Development Plans and Software Development Folders

3.1 SYSTEM REQUIREMENTS DEFINITION

During its history, the Speakeasy program has grown from a small basic R&D program to develop an advanced programmable signal processor, to a major advanced development program to build a programmable, reconfigurable, and modular architecture radio system using advanced technology to support 21st century communication requirements. The infusion of BTI funding, coupled with increased interest from the potential users, required a reexamination of the SOW. We participated in the redefinition of the Speakeasy program by supporting the program office in the following activities:

- Collection and correlation of user inputs.
- Systems Engineering evaluation of various design and technology trade-offs.
- Technical, cost, and schedule impact analysis of alternatives.
- Participation in Joint User Working Group meetings.
- Preparation and maintenance of various system documents, including:
  - Draft Systems Requirement Document
  - SOW Amendment 3
  - Draft Phase II SOW
  - Glossary and Acronym lists

3.2 SOFTWARE REQUIREMENTS ANALYSIS

A major goal of the Speakeasy program office was that the software developed in Phase I form the foundation of and be largely reused for the Phase II Speakeasy development. Requirements traceability throughout the development is key to attaining such a goal. Figure 3.1-1 shows how the contractual requirements, as stated in the SOW, are allocated among the various levels of documentation. These requirements should flow from the SOW to the System Requirements Specification (SyRS), through each section of the SyRS, then to each individual CSCI Software Requirements Specifications (SRS) or hardware specification, to the CSCI Software Design Document (SDD) or hardware design, and finally to the code modules, or hardware drawings.

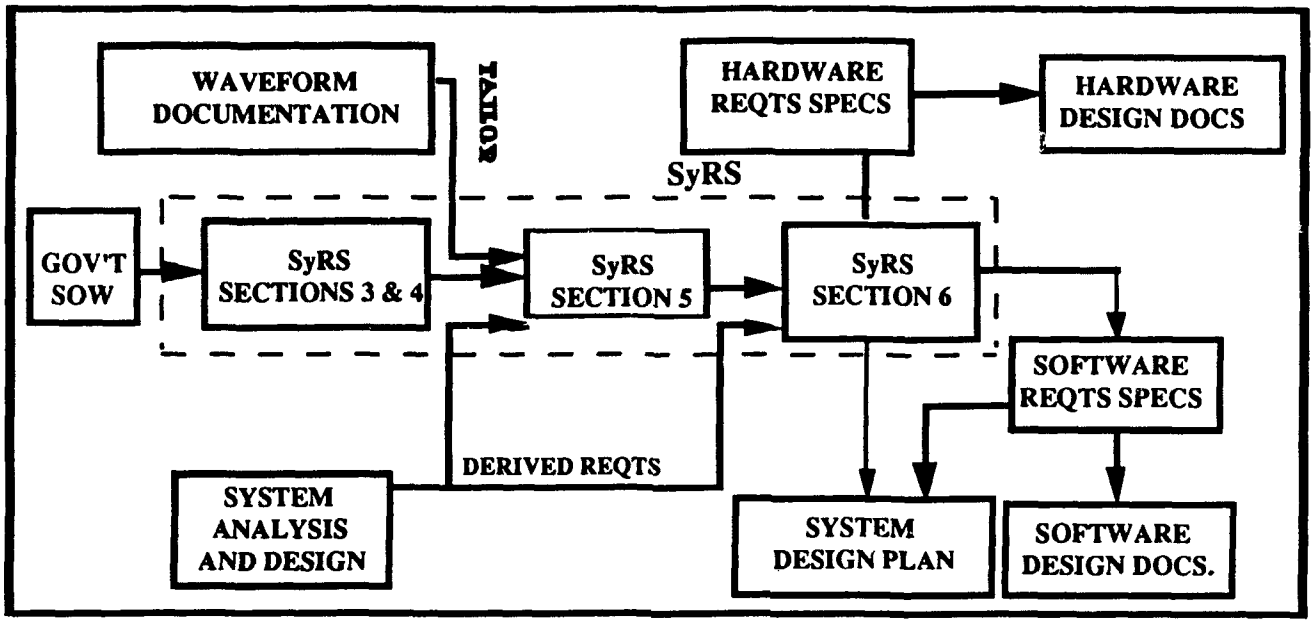

Figure 3.1-1 Requirements Traceability

TASC's assignment was to show that the delivered documentation demonstrated the traceability of requirements. Our initial assessment showed that it was impossible to determine where each system level requirement was to be met. Of 73 SOW requirements assigned to the version 5.0 SyRS, only 33 could be clearly traced into the SyRS. None of these could be traced with any certainty through SyRS into any SRS. In addition, there was little attempt to trace SRS requirements back to the SyRS. This poor quality of traceability makes any verification of requirements infeasible and tends to leave the hardware and software design in the hands of the individual engineers, with little system level integration. This would most likely result in a fragmented design, that would not meet the program technical, cost, and schedule requirements. These concerns were reported in our risk assessment.

Release 6.0 of the Speakeasy ADM System Requirements Specification (SyRS) was later reviewed for traceability from the restructured Speakeasy Statement of Work, internal traceability and consistency, and engineering soundness. In this version, the traceability from the Statement of Work (SOW) was nearly complete. The Release 6.0 SyRS captured the requirements of the SOW, as interpreted by Hazeltine, and mapped those requirements into the appropriate functional blocks (hardware, software, and interfaces) of the system. This allows allocation of these requirements to various lower level documents such as the Software Requirements Specifications (SRS). However, for many of the waveforms, the current SyRS only references the appropriate Mil-STD documents

and does not explicitly identify individual waveform requirements. This does not lay the foundation for requirements traceability to/from each of the waveform CSCI Software Requirements Specifications(SRS). The next release of the SyRS will address this issue. While there are still open issues, SyRS version 6.0 is adequate to lay the foundation of system level requirements for the Speakeasy program.

3.3 EVALUATION OF THE SOFTWARE DEVELOPMENT PROCESS

The purpose of a Software Development Plan (SDP) is to provide the Government insight into the organizations responsible for performing software development and the methods and procedures to be followed. During the contract period, we reviewed three separate revisions of each contractor's SDP, and made several site visits to each contractor to survey first hand their software development process.

3.3.1 Software Development Plan Review

We reviewed all SDPs for compliance to DoD-STD-2167A (as tailored by the SOW). Specifically, we examined their compliance to Data Item Description (DID) DI-MCCR-80030A, and their adequacy in specifying the software development management, structure, methods, and procedures to be applied in the Speakeasy software development effort. The following highlights some of the problems found, improvements that have been made and issues that continue to be addressed.

The first two versions of the Hazeltine SDP were almost entirely non-compliant to both the specific requirements and overall intention of the DID. They failed to describe Hazeltine's plans for conducting software development and provided the Government minimal insight into the software development process. The SDP described a loosely structured development that was not appropriate for the Speakeasy software. For example, the Hazeltine Ada Coding Standard was extensively tailored to make the coding easier, yet would produce code that was significantly less structured, modular, reliable and portable.

While the earlier version of the TRW SDP complied with DI-MCCR-80030A in most areas, it was generic and too broad to be of much practical value. It presented a reasonable general approach to software development, yet failed to furnish Speakeasy specific details. The SDP did not accomplish

the primary objective of describing the Speakeasy software development process in enough detail to allow the government to monitor progress and compliance.

The latest versions of the Hazeltine and TRW Software Development Plans, dated 1 December 1992 and 18 December 1992 respectively, were a major improvement over the previous SDPs. Still a few issues exist. First, both the Hazeltine and TRW SDPs referenced a number of internal company procedures, and consequently, as stand alone documents were not always specific enough in their description of techniques and procedures to be applied to the Speakeasy project. This issue is being addressed by augmentation of the current SDPs with portions of the referenced corporate documents as appendices.

Another concern that is currently being worked is that both SDPs presume complete independence from each other. The Hazeltine SDP addresses the ISC and TSC CSCIs only, while the TRW SDP addresses the MPS, Crypto Controller, and all current narrow band CSCIs. Such an approach does not support the enforcement or even consideration of project wide coding standards and practices, nor does it address global issues such as system integration and testing, configuration management, corrective action processes, etc. which should be managed at the project level.

Finally, our last major concern is that while the SDPs have evolved into well-written documents, the contractors must put the procedures and techniques that they have called out in the SDP into practice, eg. adhering to their own coding standards, the utilization of Software Development Folders etc. The objective of the 2167A requirement for an SDP goes beyond a well written document. It should be a working manual which provides software engineering guidance to the development team throughout the software development effort. We see this issue diminishing over time as the development team uses and begins to appreciate the value of the 2167A products and standards.

3.3.2 Hazeltine On-Site Surveys

During our site visits to Hazeltine, they briefed their software development process and software support functions, including CM, QA, and Testing. Their overall software management approach appears to be fairly reasonable, although overly informal for the level of software being developed ie. portable, maintainable and reusable versus a throw away prototype. Our recommendation to the Speakeasy program office is that Hazeltine and their subcontractors continue to

be encouraged to apply uniform design, documentation and coding standards across all project CSCIs, where ever practical.

Hazeltine is using the ***Adagen*** software development tool for design and coding. In Adagen, the preliminary design is object-oriented and language independent. In the Preliminary Design phase, this product assists the designer in generating Object Interaction Diagrams (OID) and State Transition Diagrams (STD). The OIDs describe how the data flows to, within, and from each object, and also provides some control information. The STDs are used to describe the various states of the system. The preliminary design is then translated (manually) into Buhr Diagrams during the detailed design phase. These diagrams are unit oriented and reflect the structure of the Ada implementation. Buhr diagrams describe the unit construction, interdependency with other units, types, function calls, and exported/imported operations. Type definitions, parameter types, comments, and other coding details can also be added. During the initial part of the coding phase, Adagen uses the Buhr Diagrams to create Ada Frames - a template of the particular unit with the structure determined from the diagram. The programmer fills in the internal details. This approach tends to greatly reduce interface errors, a common and troublesome error source, especially on large projects.

This tool provides a reasonable method to enhance the productivity and quality of the Speakeasy software. However, we raised three concerns which remain open. The first was with respect to the transition from preliminary to detailed design. Traditionally, the detailed design expands upon the preliminary design using a similar format, or with explicit cross references to the preliminary design objects. In Hazeltine's use of Adagen, however, there was no specific tracking from the preliminary design OID's to the detailed design Buhr Diagrams. Our second concern involved Hazeltine's decision that the preliminary design documents would not be maintained, i.e. changes made to the design after completion of preliminary design would not be reflected in the OIDs and STDs. Our third concern was that this tool was being used for the development of only one of Hazeltine's two CSCIs. This tool should provide a considerable improvement over a totally manual method for software design and coding, and it should be employed for the other CSCI as well.

Hazeltine's approach to CM depends upon each engineer tracking his own work, the inherent version control capability of the VAX Operating System, and a manually executed procedure. Since they have such a small development team (two software designer/programmers and a lead software engineer), this approach is feasible. Despite this, we still strongly recommended that they use an

automated version control tool, such as Source Code Control System or Revision Control System. Both of these are public domain and simple to use. The use of either of these tools would provide significant advantages:

- Less chance of losing control of the software configuration.
- A historical recreation of prior builds.
- Clearly documented changes in an easily readable format (crucial for integration).

There is unnecessary risk in using only manual methods to control the configuration of software being developed at separate sites often resulting in divergence as opposed to integration of the software.

3.3.3 TRW On-Site Survey

During our on-site visits to TRW they also presented their design methodologies and automated software development tools. TRW had a solid set of development tools for their microcode development, yet they use almost entirely manual methods in the requirements, allocation, design, documentation and testing of the Ada code. We were concerned over their lack of formal development procedures, and recommended that they investigate acquiring and using automated tools such as a requirements tracker, graphical design or case tool to support the design and code development, source code analyzer, and test/path coverage analyzer.

4. LPI/SIGNAL PROCESSING WORKSTATION ANALYSIS

As part of our efforts to enhance the RL Signal Processing Workstation, we first delivered our previously developed Vulnerability Metric software model, which is briefly described in Section 4.1. We then extended this model with the development of the Digital Density Detector module. This was delivered to Rome Laboratory on 1 November 1991, and is described in detail in Section 4.2. Finally, we did an analysis of the error probability of the variable rate waveform, computed under two different demodulation regimes: bit-by-bit decoding and symbol-by-symbol decoding. It is clear from the analysis that bit-by-bit decoding seriously limits the number of bit streams that can be added in parallel before unacceptable error rates occur. On the other hand, symbol-by-symbol decoding allows considerably more users, but at the expense of greatly increased demodulation complexity. This analysis is described in detail in Section 4.3.

4.1 THE VULNERABILITY METRIC

4.1.1 Introduction To The Vulnerability Metric

The Vulnerability Metric (VM) is a software and analysis model which determines the vulnerability of communication signals to detection by interceptors. The principal objective is to increase the speed and decrease the cost of the design and evaluation of low probability of detection (LPD) communication waveforms. The computer based technique complements field testing and reduces the reliance on expensive, few of a kind, hardware systems which take many months to build and test, and require specialized training to operate. VM offers versatility through the choices available to the user for spreading modulation, interference and noise background sources, channel models, intercept preprocessing, intercept detector architecture, operational modes, and vulnerability representations. LPD techniques are often designed to resist detection by one form of detection but may be easily detected by another. The Vulnerability Metric permits the testing, within a uniform framework, of an LPD technique against a collection of detection measures, providing an overall measure of the utility of the LPD technique. The effectiveness of detector architectures can also be compared, and development of new detector designs can be facilitated by the rapid testing capabilities of VM.

The designs of individual modules permit the user to assemble a large variety of systems quickly and easily. The Vulnerability Metric is built on the Signal Processing WorkSystem (SPW), a software product of COMDISCO Systems, Inc., and runs under OpenWindows. The Vulnerability Metric expands the SPW graphical interface to provide the user parallel access to all the capabilities of both SPW and the Vulnerability Metric. The result is a menu driven tool with a graphical interface and incorporating mathematical analysis to speed processing transparently to the user.

4.1.2 Vulnerability Concepts And Operational Modes

Vulnerability is the trade between communication performance and interceptibility, and a vulnerability metric must quantify both communication performance and intercept performance. Intercept detector performance is the ability to detect the presence of the communication signal, and depends in turn on the properties of the intercept detector's output both when the signal is present and when it is absent. A vulnerability determination therefore takes into consideration three processing sequences, one for the communicator and two for the interceptor. Receiver operating characteristic quantify detector performance as plots of the probability of detection versus probability of false alarm, while an output signal-to-noise ratio (SNR) provides an abbreviated measures of detector performance.

The Vulnerability Metric provides three modes of operation to permit the user to optimize the trade between generality and speed according to specific application needs. The modes are weak signal, white Gaussian noise; weak signal, simulated background; and non weak signal, simulated background. Table 4.1–1 summarizes the attributes of the three modes.

The weak signal, white Gaussian noise mode offers the most rapid execution by exploiting two simplifications of the vulnerability analysis. The first simplification is based on the fact that spread spectrum waveforms generally have power spectral densities that are substantially lower than the noise background. A simulation set at a single value of the input SNR can then be analytically extended to describe a range of input SNR values. The second simplification considers a background consisting of white Gaussian noise, and incorporates analytic results to reduce substantially the numerical processing time required in a vulnerability determination.

Table 4.1-1 Three Modes Of Operation Of The Vulnerability Metric

Mode	Background Type	Background Treatment	Signal Present Simulation	Signal Absent Simulation	Variables In Output SNR Plot
Weak signal, white Gaussian noise	White Gaussian noise	Analytic	Signal	Noise	Input SNR Collect Time
Weak signal, simulated background	General noise interference	Simulation	Signal	Background	Input SNR Collect Time
Nonweak signal, simulated background	General noise interference	Simulation	Signal Plus Background	Background	Collect Time

The choice of metric block from the Metrics menu is dictated by the choice of operational mode. In the weak signal, white Gaussian noise mode, the Weak Signal metric must be chosen. The resulting graphical output gives the output SNR for varying input SNR and collect time, with the collect time dependence expressed in terms of the number of collected symbols. Changing the data rate or other system parameters requires a generating a new plot. The upper limit of displayed input SNR is chosen to ensure that the weak signal approximation is good everywhere in the plot. Increasing the spreading, the ratio of bandwidth to data rate, will increase the range of consistent input SNR.

The weak signal, simulated background mode can deal with more general backgrounds such as static interference and nonwhite or non-gaussian noise, and requires simulation of both the signal and the background. This mode uses the same metric block, Weak Signal, and produces the same form of graphical output as the weak signal, white Gaussian noise mode.

The non weak signal, simulated background mode provides the most generality and is not restricted to either weak signals or a Gaussian white background. The Simulated Output SNR block must be used as the metric block, and the graphical output shows output SNR as a function of collect time. The collect time dependence is expressed in terms of the number of collected symbols. Changing the data rate or other system parameters requires generating a new plot.

Once the Vulnerability Metric is installed, typing "vm" at an Open Windows prompt starts VM and produces a start up screen which contains the Components Menu, the System Menu, and the System Viewport. The Components Menu is the source of blocks which the user selects to build a system in the System Viewport. The System Menu allows the user to save and execute a configured system. Vulnerability Metric results are displayed in a separate graphics window and can be directed to a PostScript printer.

The Components Menu is in the upper right corner of the window and provides hierarchical access to the top level building blocks of the system. Components are divided into five types:Waveform, Background, Channel, Preprocessor, Detector, and Metric, and the user assembles systems by selecting individual components of each type from a submenu. Right clicking on a menu option either selects that item for insertion into a system or brings up a submenu if further options are available. Clicking on the title of a submenu returns the user to the top level of the Components Menu

Switching from the Vulnerability Metric to SPW and back is also accomplished through the menus. Selecting the Components Menu heading in the Vulnerability Metric brings up the SPW Block menu, and selecting the "VULN METRIC" button in the Block menu recovers the Vulnerability Metric menus.

A User Guide will be developed to provide full details of the Vulnerability Metric menus; the individual components; and the parameters that configure components and systems.

4.2 THE DIGITAL DENSITY DETECTOR

4.2.1 Introduction

The Digital Density Detector (DDD), as described in Reference 1, has been custom-coded and integrated as a pre-processor into our Vulnerability Metric software package, described in Section 4.1.

The DDD is an adaptive detector structure for known signals in unknown noise, and can provide considerable gain over simple matched filter detection, if the noise characteristics deviate considerably from Gaussian. This is especially true in the presence of interference, as demonstrated in this section. The performance of the system has been tested against a number of noise and interference distributions, and appears to be quite good in all cases. The flexibility of the SPW environment allows fairly rapid testing and evaluation of sensitivity to algorithmic changes and to parametric variation.

4.2.2 Description Of The Algorithm

Suppose the presence or absence of a known deterministic (discrete-time) signal is required to be tested, given a received signal consisting either of the sum of signal and noise or just the noise alone. Further assume that the noise samples are independent from sample to sample and identically distributed. That is, the received signal is

$$r_k = \theta s_k + n_k \qquad k = 0, \ldots, N \tag{4.2-1}$$

where θ is either zero or one, and s_k is the known signal sequence.

The optimum test, either to minimize the probability of error or in the Neyman-Pearson sense, consists of computing the *likelihood ratio* defined by

$$l = \sum_k log \left\langle \frac{p_n(r_k - s_k)}{p_n(r_k)} \right\rangle \tag{4.2-2}$$

where $p_n()$ is the probability density function of the noise variates, and comparing to a suitable threshold.

If the noise is Gaussian, the likelihood test reduces to the well known correlation receiver, which forms

$$l = \sum_k r_k s_k \,. \tag{4.2-3}$$

The correlation receiver is commonly used in non-Gaussian problems, either because the exact noise statistics are unknown or the because the optimum decision statistic is considerably more complicated than the simple correlator. In Reference 2, it was demonstrated that a low-signal approximation to (4.2-2) may be obtained from the Taylor series approximation to $p_n(\ )$. If s_k is small,

$$p_n(r_k - s_k) \cong p_n(r_k) - s_k \frac{d}{dr} p_n(r_k) \tag{4.2-4}$$

and (4.2-2) becomes

$$l \cong \sum_k \log \left\langle 1 - s_k \frac{\frac{d}{dr} p_n(r_k)}{p_n(r_k)} \right\rangle. \tag{4.2-5}$$

Since log (1+x) =x for x small, (4.2-5) may be approximated as

$$l \cong - \sum_k \frac{s_k \frac{d}{dr} p_n(r_k)}{p_n(r_k)}, \tag{4.2-6}$$

so that the small signal approximation to the optimal receiver may be thought of as a correlator acting on a non-linearly processed version of the received signal. This allows (4.2-6) to be rewritten as

$$l \cong \sum_k s_k g\ (r_k) \tag{4.2-7}$$

where

$$g(x) = - \frac{d}{dx} \log (p_n(x)). \tag{4.2-8}$$

Note that by substituting the Gaussian distribution into (4.2-8), one again obtains the correlation receiver.

The Digital Density Detector described in Reference 1 implements the structure of (4.2-7) adaptively in the absence of knowledge about the noise distribution. This is particularly important in the presence of jamming, the statistics of which may be unknown or even time-varying. The approximation is accomplished in the weak signal case by estimating the distribution of the noise over the observation interval using a histogram. If the $\{s_k\}$ have zero mean, than the error in approximating the noise distribution by a histogram taken in the presence of signal is quadratic in the $\{s_k\}$. Note that a small signal is required for two reasons: first, the Taylor series approximation is only valid for small signals, and second, a large signal will contaminate the estimate of the noise distribution. One might reason that in the large signal case performance of the simple correlation receiver is sufficiently good, and that it would be unnecessary to attempt to implement a decision statistic closer to optimum. Alternatively, one could embed the DDD in a decision feedback structure, which might be expected to work well at both low and high SNR's. Such a structure would require quite a bit of storage, and would assume that the noise statistics vary slowly from frame to frame, but the improved performance might be justified in some cases.

Simply summarized, the Digital Density Detector is implemented by performing a histogram over the desired time interval and using a first order difference applied to the logarithm of the histogram bin counts to approximate the derivative used in (4.2-8). In the bandpass case, both signal and noise are generally treated as complex quantities, but the one dimensional result is readily generalized to the complex case in a direct manner. Furthermore, if the noise distribution is assumed to be radial symmetric and that the signal is subject to an unknown or uniformly distributed phase shift, then the two dimensional signals may be converted to polar notation. Then, noting that there is no dependence on angle, the nonlinear processing denoted by $g(\,)$ may be achieved by an operation in the radial dimension.

The theoretical behavior of DDD for complex signals is derived in Reference 1, but that report contains some errors in the passage from rectangular to polar coordinates. Our implementation and analysis depend upon a corrected derivation of the fundamental equations. The important relations to note are that if the radius of the received signal is acted upon by a non-linearity $g(\,)$ the quantities

$$K_M = -\frac{1}{2}\int_0^\infty g(r)\,\frac{d}{dr}\left(\frac{1}{r}p_R(r)\right) r\,dr \qquad (4.2\text{-}9)$$

and

$$K_v = \frac{1}{2}\int_{-\infty}^{\infty} g^2(r)\, p_R(r)\, dr \tag{4.2-10}$$

define the performance of the detector, in that the output signal to noise ratio is given by

$$SNR_g = \frac{S^2 K_M^2}{K_v}, \tag{4.2-11}$$

where S^2 is the magnitude squared of the signal. If the variance of a one-dimensional noise component is σ_n^2, then the signal to noise ratio of a linear detector is given by

$$SNR_{linear} = \frac{S^2}{\sigma_n^2} \tag{4.2-12}$$

and the gain in using DDD rather than a linear detector is given by

$$G = \frac{K_M^2}{K_v}\sigma_n^2. \tag{4.2-13}$$

It can be readily shown that G is maximized by choosing $g(\,)$ to be

$$g(r) = \frac{d}{dr}\log\left[\frac{1}{r}p_R(r)\right]. \tag{4.2-14}$$

This function is approximated by estimating $p_r(\,)$ by accumulating a histogram of values of the complex magnitude of the received signal. In the small signal case, the histogram is assumed to be an accurate estimate of the distribution of the magnitude of the noise. An approximation to the function in (4.2-14) is gotten by taking first order differences of the appropriate function of the histogram values.

4.2.3 Performance Of Simulated System

The Digital Density Detector has been implemented by a custom-coded block and integrated with the Spread Spectrum Vulnerability Metric software being developed on SPW. The performance of the detector has been tested on three known noise distributions for whom the theoretical performance gain may be calculated, and on a system with noise and interference.

The three noise distributions tested include the two dimensional Gaussian distribution and two distributions designed to have heavier tails than the Gaussian. These distributions are radial versions of the Laplace and Cauchy densities. In all cases the noise is assumed to be radially symmetric, so that the distribution of the angle of the noise is always given by

$$p_{\Theta}(\theta) = \frac{1}{2\pi} \quad -\pi \leq \theta \leq \pi \tag{4.2-15}$$

The radial distribution when the noise is Gaussian is given by

$$P_R(r) = \frac{r}{\sigma^2} e^{\frac{-r^2}{2\sigma^2}} \tag{4.2-16}$$

where the variance of a single noise component is σ^2.

When then radial distribution decays exponentially as the first power of the radius, with a *density given by*

$$p_R(r) = \lambda^2 r e^{-\lambda r} \tag{4.2-17}$$

the variance of a single noise component is $3/\lambda^2$, and we shall refer to this case as the Gamma distribution.

In the third case the distribution we use will be given by

$$p_R(r) = \frac{1}{\alpha} \frac{x/\alpha}{\left[\sqrt{1+(x/\alpha)^2}\right]^3} \tag{4.2-18}$$

This is a two-dimensional Cauchy distribution, in that the individual components are marginally Cauchy, and the variances are infinite.

In addition to these three cases, which are capable of being analyzed exactly, we simulated the case where additive Gaussian noise and a single strong interferer are present. While this last case may not be analyzed in closed form, the performance gain afforded by using DDD is striking as will be seen in Section 4.2.4. We now discuss the three cases introduced above.

Gaussian Noise

When the noise is Gaussian the optimum receiver is the linear correlator. The constants K_V and K_M have the common value

$$K = \frac{1}{\sigma^2}. \tag{4.2-19}$$

When the signal sequence is of length N and the s_k have a common amplitude of s, as in direct sequence spread spectrum signaling, then the post-processing SNR is given by

$$SNR = \frac{S^2}{\sigma^2}N\ . \tag{4.2-20}$$

The DDD should adaptively approximate the linear correlator, and its performance approach (4.2-20), for large enough N.

Gamma Noise

When the noise has the gamma distribution given by (4.2-17), $-K_M$ and K_V are given by

$$K = \frac{\lambda^2}{2}. \tag{4.2-21}$$

The SNR for a linear receiver may be shown to be given by

$$SNR_1 = \frac{\lambda^2}{3}, \tag{4.2-22}$$

so that the gain in going from a linear detector to the DDD is a factor of 3/2 or 1.8 db.

The optimum radial nonlinearity for gamma noise is given by

$$g(r) = \lambda\,, \tag{4.2-23}$$

that is, each received complex variate is mapped to a point with the same phase but constant magnitude before combining.

Cauchy Noise

When the radius of the noise is given by (4.2-18), the noise coordinates are marginally Cauchy, i.e. the x coordinate has a density given by

$$p_x(x) = \frac{1}{\pi\alpha\,[1 + (x/\alpha)^2]}\,, \tag{4.2-24}$$

where α is a scaling parameter.

The variances of the marginal and radial distributions are infinite, so that the signal to noise ratio is $-\infty$ db. Non-trivial detection and false alarm probabilities are not precluded, however, since the distribution itself is well defined. The Cauchy distribution has the remarkable property that the normalized sum of Cauchy variates has the same distribution as the individual variates (i.e. no law of large numbers applies), so that there is no processing gain available from linearly combining measurements. That is, the linear correlator works as well or as poorly as a receiver which merely tests a single sample against a threshold. It is here that the advantage, indeed the necessity, of a nonlinear processor like DDD becomes apparent. The optimum low signal radial non-linearity is given by

$$g(r) = \frac{3x}{\alpha^2 + x^2}\,. \tag{4.2-25}$$

The signal to noise ratio is found to be

$$SNR = \frac{3}{5\alpha^2}\,. \tag{4.2-26}$$

One might notice the change in behavior of $g(\ )$ as the tails of the noise distribution get progressively heavier: in the Gaussian case, $g(\ )$ is linear. When the noise distribution tails off exponentially $g(\ )$ is a constant. When the distribution's decay is merely algebraic, $g(\ )$ ceases to even be monotonically increasing.

4.2.4 Simulated Results

The figures that follow summarize the behavior of the simulated DDD in additive Gaussian noise, gamma noise, and Gaussian noise plus a single BPSK interferer. There is no accompanying figure for Cauchy noise, since the variance for the linear receiver is infinite in this case. The DDD system is implemented as a non-linearity applied to the magnitude of the received signal. This radial implementation is shown in Reference 1 to be equivalent to the two-dimensional implementation.

In all cases, histogram binning was done using 50 bins, but the best clipping level depends upon the noise level. If the clipping level is too low, the received datum is clipped too often, while if it is too high, the density function of the noise is too coarsely estimated. For signal to noise ratios around -30 or -20 db (i.e. noise variances on the order of 100 to 1000) clipping at a radial value of 1000 was used. For SNR of -40 db (noise variance of the order of 10000), this was inadequate, but doubling the clipping point to 2000 sufficed to provide near-optimum performance. The effect of too coarse a quantization was not observed in this simulation.

Signal to thermal noise ratios from -40 to 0 db are simulated, and the output signal to noise ratio after de-spreading a spread spectrum signal with 30 db processing gain is plotted against input SNR.

Results for the two noise cases, for which the problem is analyzable, are very close to those predicted by theory: for the Gaussian case the resulting SNR is the same as a matched filter, which is theoretically 27 db above the input (30 db processing gain less 3 db for the incoherent detector). In the gamma case the optimum processor is 1.8 db better than the matched filter, hence output SNR should be 28.8 db above the input. The simulated results, shown in Figures 4.2-1 and 4.2-2 respectively, are exactly where they should be.

In the case of additive noise and an interferer, the interferer is broadband (BPSK keyed at around 500 Hz, while the spread spectrum chip rate employed is 1000 Hz.) and 30 db above the signal. Since the interferer is broadband, the de-spreading operation alone is largely ineffective. Indeed, the interference acts essentially the same as white Gaussian noise, in that the matched filter

will work as though there were additive noise with power of 30 db above carrier plus the thermal noise. In particular the output SNR saturates at 0 db, performance which is verified in Figure 4.2-3. The striking advantage of the DDD system is evident in this case, since as the thermal noise gets lower, the DDD processor obtains an output SNR which is within 2-3 db of the performance that the matched filter would obtain if the interferer were not even present

As indicated in Reference 1, in the presence of an interferer who mimics Gaussian noise, or when a large number of interferers result in Gaussian-like behavior due to a central limit theorem effect, the DDD is far less effective. Clearly, however, when the noise statistics deviate significantly from Gaussian, as in the cases presented here, DDD allows considerable gain over the linear correlator.

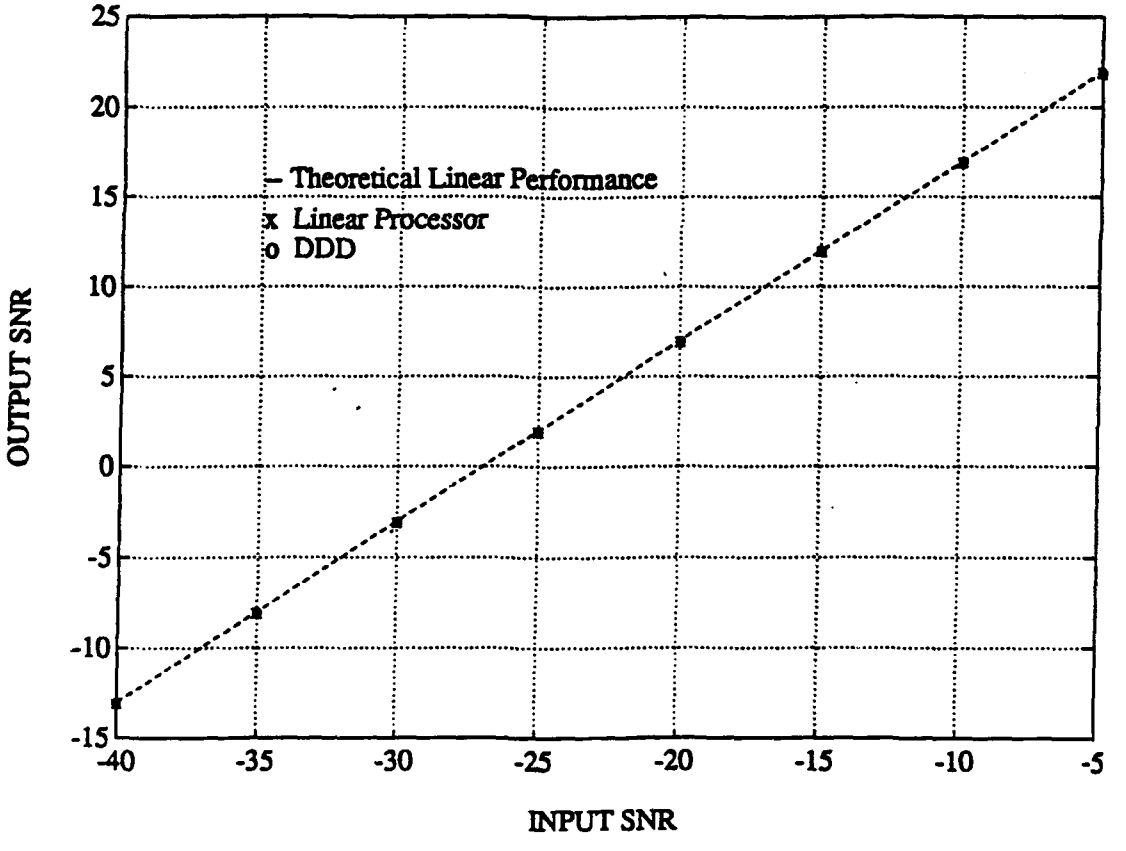

Figure 4.2-1 Performance In Gaussian Noise

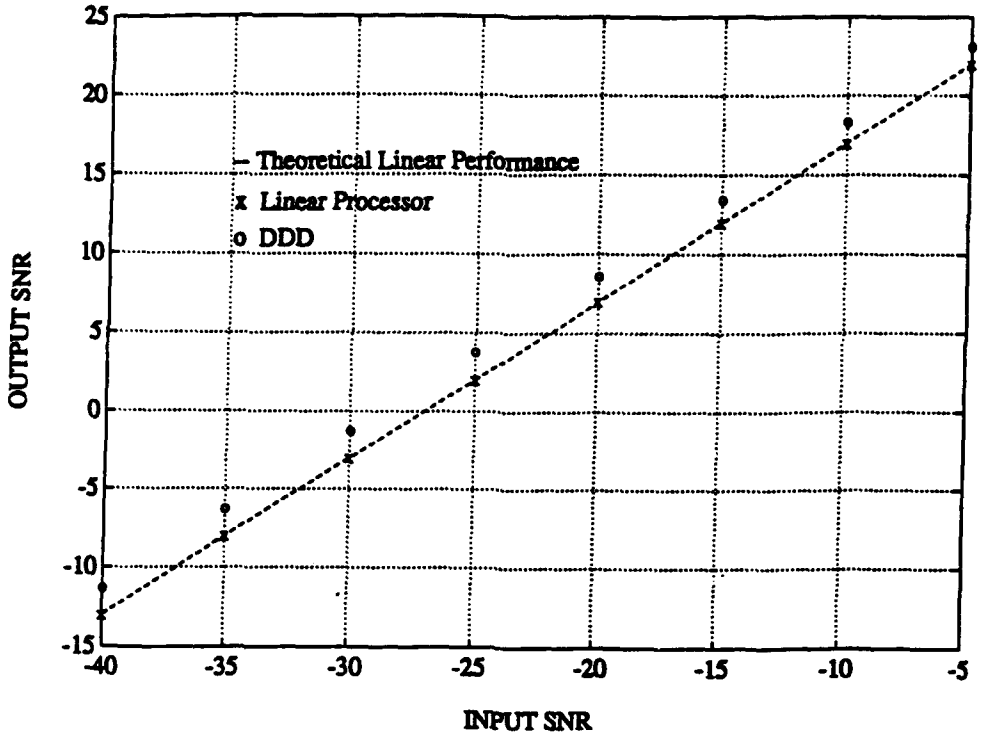

Figure 4.2-2 Performance In Gamma Noise

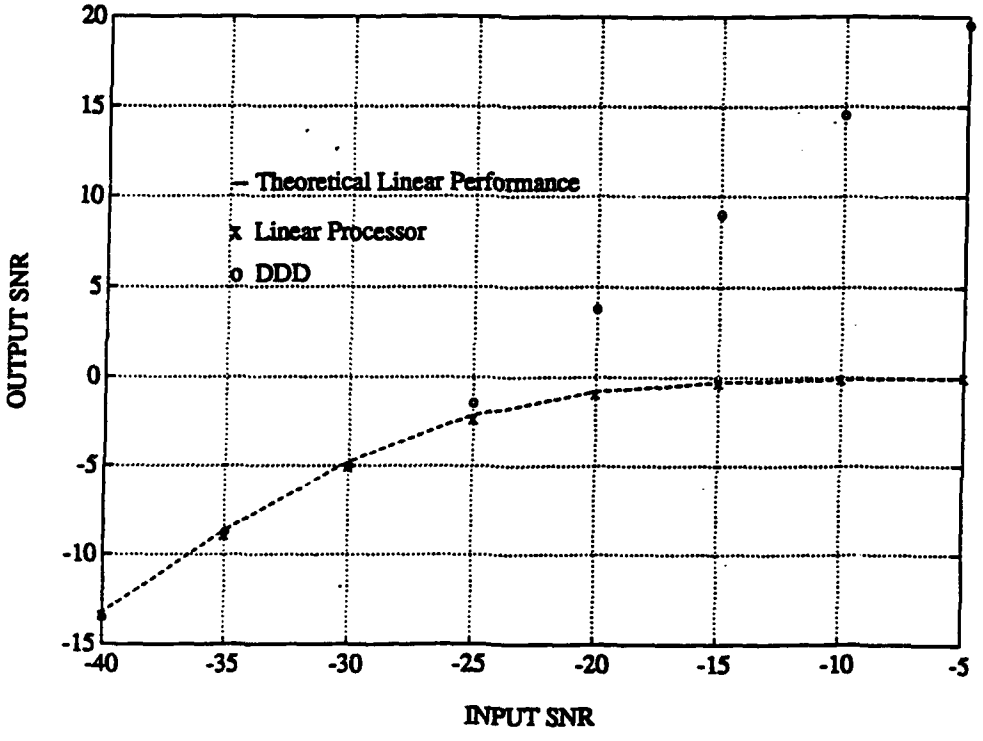

Figure 4.2-3 Performance In Interference

4.3 VARIABLE RATE MODULATION SCHEME

The variable rate modulation scheme allows a range of data rates to be supported by a single waveform. The only difference between the minimum and maximum rates is in the generation of the original bit stream that drives the modulator.

The waveform that is generated is the sum of a collection of independent identically distributed waveforms, and as such is approximately Gaussian distributed. This feature is advantageous in an environment in which low probability of detection (LPD) must be maintained, because detectors that depend upon the non-Gaussian statistical nature of man-made signals, for example receivers using polyspectral techniques, will be thwarted by approximately Gaussian waveforms. Combining the use of the variable rate waveform with chip rate filtering produces a waveform which at least as far as quadratic detectors are concerned strongly resembles wideband Gaussian noise.

Assuming that an energy per bit of E_b is desired and N chips per symbol are available, the variable rate modulation scheme works as follows: to generate each transmitted symbol of N chips, generate km individual independent binary sequences, each of length N. To transmit k bits, divide the sequences into k groups of m sequences each, with each group assigned to a particular bit. To transmit the data multiply each stream in the i^{th} subgroup by the i^{th} bit and add all of the sequences together. The resultant waveform is normalized to power kE_b by multiplying by $\sqrt{E_b/mN}$ (assuming that the original sequences take on values $\pm$ 1). As a benchmark, the bit error probability of antipodal signaling in the presence of additive white Gaussian noise with power spectral density $N_0/2$, corresponding to the case where m and k are 1 is given by

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{4.3-1}$$

In the remainder of this chapter, error results for the three types of demodulation will be obtained, for comparison with (4.3-1).

Figure 4.3-1(a - c) summarizes the structure of the multirate modulator, and a comparison for K = 4, and k = 1, 2, and 4. Clearly the only difference between the three cases is the number of sources and the switching software. Key generation for the spreading sequences, and the appearance and statistics of modulated waveform are the same regardless of k.

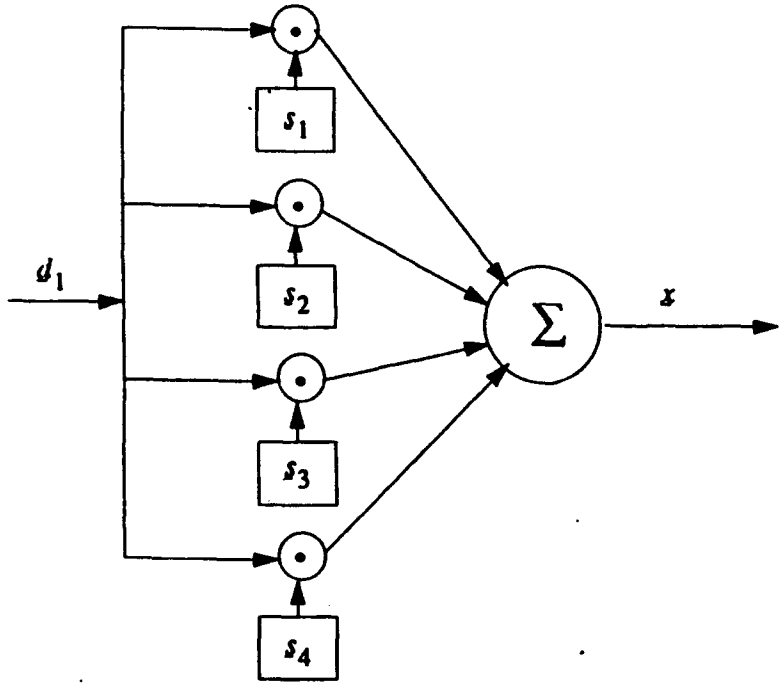

Figure 4.3-1a Single Data Stream

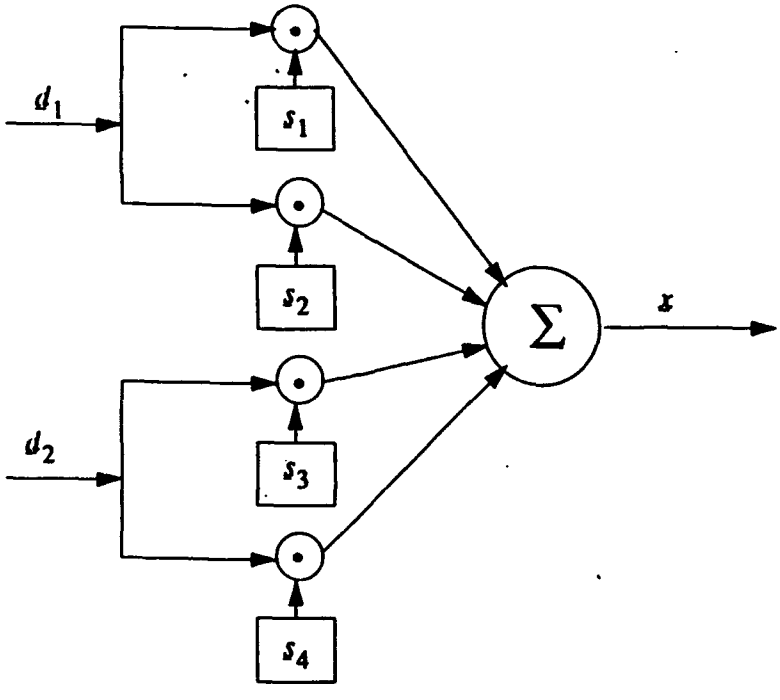

Figure 4.3-1b Dual Data Stream

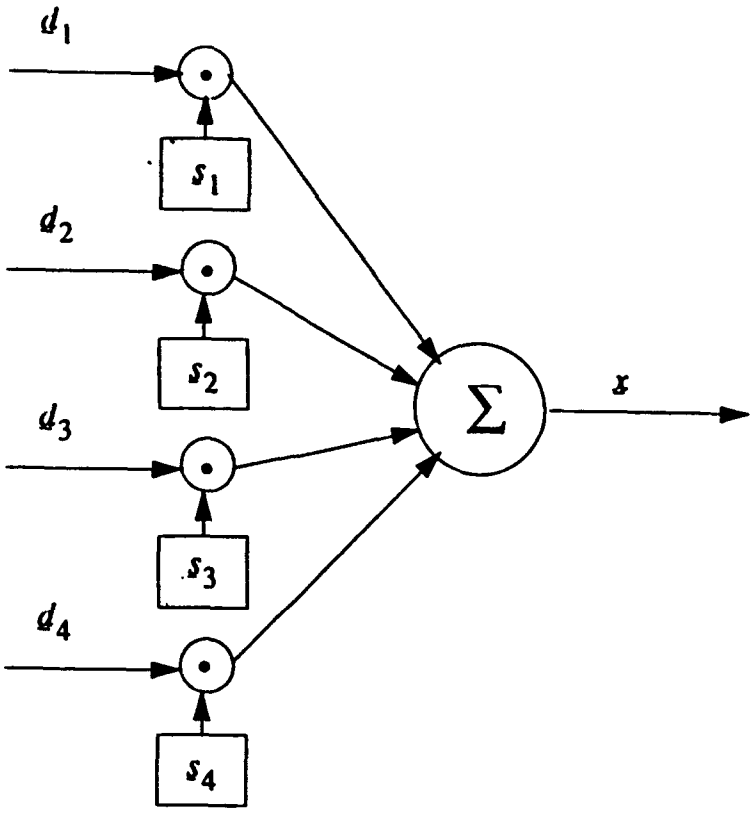

Figure 4.3-1c Quadruple Data Stream

4.3.1 Bit-by-bit Decoding

In this section we describe bit-by-bit decoding and derive its probability of error. Assume the nomenclature presented in Appendix A. That is, exploiting the natural isometry between piecewise constant waveforms and N chips and N- vectors, denote the set of generated PN sequences by $\{x_i\}$. Let the j^{th} bit be transmitted by the waveform.

$$s_i = \sqrt{T} \sum_{k=1}^{m} x_{im+k} \tag{4.3-2}$$

where, T is an arbitrary constant, so that the transmitted waveform corresponding to a transmitted binary data vector $\underline{b}$ is represented by

$$t(\underline{b}) = \sqrt{T} \sum_{i=1}^{k} b_i s_i. \tag{4.3-3}$$

An individual bit, for example the k^{th} bit is demodulated by forming the correlation between the received waveform, which is equal to $t(\underline{b})$ corrupted by additive noise, and s_k, the waveform corresponding to that bit. The sign of the resulting statistic determines whether a *0*, corresponding to a negative value, or a 1, corresponding to a positive value, was sent.

Suppose that we are demodulating the first bit, denoted by b_1, and assume that a 1 was sent. In this case

$$t(\underline{b}) = \sqrt{T} \left(s_1 + \sum_{i=2}^{k} b_i s_i \right). \tag{4.3-4}$$

The received signal is then

$$r = \sqrt{T} \left(s_1 + \sum_{i=2}^{k} b_i s_i \right) + n, \tag{4.3-5}$$

where n is an N - vector of independent identically distributed Gaussian random variates, each with variance assumed to be $N_0/2$. The decision statistic for b_1 is then given by

$$l_1 = r^T s_1, \tag{4.3-6}$$

which by substituting (4.3-5) is

$$l_1 = \sqrt{T} |s_1|^2 + \sqrt{T} \sum_{i=2}^{k} b_i s_i^T s_1 + n^T s_1. \tag{4.3-7}$$

By construction, the individual bits and components are independent and identically distributed, and we will assume that N is large enough so that the second term of (4.3-7) can be assumed to be Gaussian. We may therefore, estimate the probability of error by computing the mean and variance of l_1, and using the appropriate tail of the error function.

The means of the second and third terms of (4.3-7) are zero, and the mean of the first term can be computed by applying (A-6) of the Appendix, to yield

$$E\left[l_1\right] = \sqrt{TE}\left[\left|s_1\right|^2\right] = \sqrt{T}\, mN. \tag{4.3-8}$$

As for the variance, the square of l_1 is given by

$$l_1^2 = T\left|s_1\right|^4 + T\sum_{i=2}^{k}\sum_{j=2}^{k} b_i b_j s_i^T s_1 s_1^T s_j + n^T s_1 s_1^T n + Crossterms. \tag{4.3-9}$$

The expected values of the cross terms are all zero. The mean of the first term is found by applying (A-19). The means of the second and third terms can be found by exploiting the independence of s_i and s_j for $i \neq j$, the independence of the transmitted bits from waveforms, and both from the additive noise, and noting that

$$E\left[s_1 s_1^T\right] = E\left[\sum_{i=1}^{m}\sum_{j=1}^{m} x_i x_j^T\right] = mE\left[x_i x_i^T\right] = mI \tag{4.3-10}$$

where I is the NxN identity matrix. Finally

$$E\left[l_1^2\right] = TN\,(N+2)\,m\,(m-1) + TmN^2 + Tm^2\,(k-1)\,N + m\frac{N_0}{2}. \tag{4.3-11}$$

Subtracting the square of (4.3-8) to obtain the variance yields

$$Var\,(l_1) = 2TNm\,(m-1) + Tm^2\,(k-1)\,N + Nm\,\frac{N_0}{2}. \tag{4.3-12}$$

Now the average energy per bit, denoted by E_b, is $E\left[s_i^2\right] = TmN \;\; = TmN$, so

$$Var\,(l_1) = 2E_b\,(m-1) + mE_b\,(k-1) + Nm\frac{N_0}{2}. \tag{4.3-13}$$

and the per bit signal to noise ratio, given by

$$\gamma_b = \frac{(E[l_1])^2}{Var(l_1)} \tag{4.3-14}$$

is

$$\gamma_b = \frac{E_b^2/T}{mN\frac{N_0}{2} + 2E_b(m-1) + E_b m(k-1)} = \frac{E_b^2}{E_b\frac{N_0}{2} + 2E_b Tm(1-\frac{1}{m}) + E_b mT(k-1)}, \tag{4.3-15}$$

which may be simplified to

$$\gamma_b = \frac{E_b}{\frac{N_0}{2} + \frac{E_b}{N}(k+1-\frac{1}{m})}, \tag{4.3-16}$$

so that the probability of error may be approximated by

$$P_b \approx Q\left(\sqrt{\frac{2E_b/N_0}{1+2E_b/N_0(\frac{k+1-2/m}{N})}}\right). \tag{4.3-17}$$

The denominator of (4.3-17) may be divided into three terms. The first term is conventional additive Gaussian noise. The second and third are proportional to E_b/N_0 and consist of a self noise and cross noise terms from the signal itself. The self noise term is

$$n_{self} = 4\frac{E_b}{N_0}\frac{(1-\frac{1}{m})}{N}. \tag{4.3-18}$$

This term is caused by random fluctuations in the signal energy used to transmit any given bit, owing to random correlations between the x_i. Observe that if $m = 1$, the signal energy is deterministically equal to E_b, and the self noise term is absent. This term shows that performance actually degrades modestly as m increases, since the self noise increases as the number of sequences driven by a given bit increases. Recall that the purpose of using a large number of sequences is to

approximate Gaussian noise, and thus to enhance covertness, not to improve communication performance. Fully half of the total loss is incurred in going from a single sequence to two.

The remaining cross noise term due to random correlations between the codewords assigned to different bits is given by

$$n_{cross} = 2\frac{E_b}{N_0}\frac{(k-1)}{N}, \tag{4.3-19}$$

and is directly proportional to the number of interfering sequences.

Note that as $E_b/N_0 \longrightarrow \infty$, the error probability does not go to zero as in the case of a single deterministic waveform, but approaches

$$P_\infty = Q\left(\sqrt{\frac{N}{k+1-\frac{2}{m}}}\right), \tag{4.3-20}$$

so that for k greater than a fairly modest fraction of N, unacceptable irreducible error probability occurs.

We plotted the performance of bit-by-bit decoding for a processing gain of 24 db (N = 256), and a range of data rates. Results showed that for even 64 bits per symbol, that is operation at roughly 25% of capacity, unacceptable irreducible error rate is obtained. In addition, even at an error rate of 10^{-3}, with only 16 bits per symbol, a degradation of roughly 5 db is incurred.

4.3.2 Symbol-By-Symbol Decoding

In symbol-by-symbol decoding a likelihood function is computed for each possible combination of transmitted bits, and the bit pattern corresponding to the largest is chosen. As in any coded or modulated system, the probability error depends upon the set of Euclidean distances between code words. In the probabilistic setting we describe analysis is not so simple, and we will use a second order analysis that depends upon the results of the Appendix A.

Suppose that the binary vector $\underline{b}$ is transmitted, so that the received waveform is

$$\underline{r} = \sum_{i=1}^{k} b_i \underline{s}_i + \underline{n}. \tag{4.3-21}$$

The log likelihood function in Gaussian noise for an arbitrary bit vector $\underline{b}'$ is readily shown to be

$$l(\underline{b}') = 2\underline{t}^T(\underline{b}')\,\underline{r} - |\underline{t}(\underline{b}')|^2, \tag{4.3-22}$$

so that in particular the difference between the likelihood function of the true bit vector $\underline{b}$ and an arbitrary different vector $\underline{b}'$ is

$$l(\underline{b}) - l(\underline{b}') = 2\,(\underline{t}(\underline{b}) - \underline{t}(\underline{b}'))^T \underline{r} + |\underline{t}(\underline{b}')|^2 - |\underline{t}(\underline{b})|^2. \tag{4.3-23}$$

Noting that $\underline{r} = \underline{t}(\underline{b}) + \underline{n}$, (4.3-23) becomes

$$\Delta = l(b) - l(b') = |t(b) - t(b')|^2 + 2\,(t(b) - t(b'))^T n. \tag{4.3-24}$$

Now the statistics of Δ depend upon the statistics of $\underline{t}(\underline{b}) - \underline{t}(\underline{b}')$. Clearly, the statistics of the latter difference depend upon only the number of places in which $\underline{b}$ *and* $\underline{b}'$ differ. Let us suppose that $\underline{b}$ *and* $\underline{b}'$ differ in J bits. We may assume these to be the first J bits, so that we can assume that $\underline{b}$ is the all 1's vector, and $\underline{b}'$ is -1 in the first J places, so

$$\underline{t}(\underline{b}) - \underline{t}(\underline{b}') = 2\sum_{i=1}^{J} \underline{s}_i. \tag{4.3-25}$$

Now

$$E[\Delta] = E[\,|\underline{t}(\underline{b}) - \underline{t}(\underline{b}')|^2] \tag{4.3-26}$$

and

$$E\left[\,|t(b)-t(b')|^2\right]=4\sum_{i=1}^{J}\sum_{j=1}^{J}s_i^T s_j=4\underline{v}^T\underline{v}, \tag{4.3-27}$$

where

$$\underline{v}=\sum_{i=1}^{mJ}x_i. \tag{4.3-28}$$

Applying (A-6)

$$E\left[\,|t(b)-t(b')|^2\right]=4mJNT. \tag{4.3-29}$$

As for the variance of Δ,

$$E\Delta^2=|\,t(b)-t(b')\,|^4+4|t(b)-t(b')|^2\,(t(b)-t(b'))^T n+4\,(t(b)-t(b'))^T nn^T(t(b)-t(b')) \tag{4.3-30}$$

and

$$E\left|\Delta^2\right|=\left[\,|E\,t(b)-t(b')|^4\right]+4E\left[(t(b)-t(b'))^T nn^T(t(b)-t(b'))\right]. \tag{4.3-31}$$

Since the noise is white with variance $\frac{N_0}{2}$, (4.3-31) becomes

$$E\left[\Delta^2\right]=E\left[\,|t(b)-t(b')|^4\right]+4\frac{N_0}{2}E\left[\,|t(b)-t(b')|^2\right].$$

Thus the variance of Δ, given by

$$Var\,(\Delta)=E\left[\Delta^2\right]-E^2\left[\Delta\right], \tag{4.3-32}$$

is seen to be given by

$$Var\,(\Delta) = Var\,(\,|\,\mathcal{I}(b) - \mathcal{I}(b')\,|^{\,2}\,) + 4\,\frac{N_0}{2}\,E\big[\,|\,\mathcal{I}(b) - \mathcal{I}(b')\,|^{\,2}\big]. \tag{4.3-33}$$

By applying (A-21) to the first term, and (A-6) to the second term, we find

$$Var\,(\Delta) = 16\,(2NT^2mJ\,(mJ-1)) + 16\frac{N_0}{2}mJNT. \tag{4.3-34}$$

Since $mNT = E_b$, (4.3-34) may be written as

$$Var\,(\Delta) = 32\frac{E^2_b}{N}J\,(1-\frac{1}{mJ}) + 16\,\frac{N_0}{2}JE_b, \tag{4.3-35}$$

and the net signal to noise ratio given by $\dfrac{E^2[\Delta]}{Var\,(\Delta)}$, is

$$SNR = \frac{16J^2E_b^2}{32\frac{E^2_b}{N}J\,(1-\frac{1}{mJ}) + 16\frac{N_0}{2}JE_b} = \frac{2JE_b/N_0}{\frac{4E_b/N_0}{N}J\,(1-\frac{1}{mJ}) + 1}. \tag{4.3-36}$$

Since the number of b' that differ from $\underline{b}$ in exactly J places is $\binom{k}{j}$, the union bound may be applied to estimate the bit error probability by

$$P_b \approx \sum_{J=1}^{k} \frac{J}{k}\binom{k}{J} Q\left(\sqrt{\frac{2JE_b/N_0}{\frac{4E_b/N_0}{N}J\,(1-\frac{1}{mJ}) + 1}}\right). \tag{4.3-37}$$

Note the similarity of the argument of the Q ( ) function to (4.3-17) except that a gain of a factor of J occurs.

If (4.3-17) is evaluated in the limit as $E_b/N_0 \rightarrow \infty$, then actually it approaches $2^{k-1} Q(\sqrt{\frac{N}{2}})$, which like (4.3-20) indicates an upper limit of around $N/4$ for acceptable performance. In this case, however the expression for bit error probability is pessimistic, owing to the use of the union bound. If we condition the expression for the error event upon the number of bits in error and exploit the fact that probabilities are always less than or equal to one, we obtain the better bound

$$P_b \approx \sum_{J=1}^{k} \frac{J}{k} min\left(\binom{k}{J} Q\left[\sqrt{\frac{2JE_b/N_0}{\frac{4E_b/N_0}{N} J\,(1-\frac{1}{mJ}) + 1}} \right], 1 \right). \qquad (4.3\text{-}38)$$

The estimate of error probability given by (4.3-38) was plotted for the same range of parameters that were evaluated for bit-by-bit decoding in the previous section. Performance is remarkably improved over bit-by-bit decoding in this case, since even at 64 bits per symbol, there is asymptotically no degradation in performance compared to data at one bit per symbol. Unfortunately, the loss of accuracy introduced by the union bound precludes analytic evaluation beyond 64 bits per symbol, though obviously a substantially higher bit rate could be supported with no loss in performance. Simulation would presumably bear this out.

Naturally this improvement in performance is obtained at the expense of dramatically increased complexity: while the complexity of bit-by-bit decoding grows linearly as the number of bits increases, the complexity of symbol-by-symbol decoding grows exponentially.

4.4 CONCLUSIONS

The DDD described in Reference 1 has been incorporated into the Vulnerability Metric software model. The modularity and extensibility of the VM provides an environment in which the capabilities and limitation of innovative receiver structures such as the DDD are readily examined.

The ease with which test environments involving a variety of noise statistics and interference backgrounds can be generated has allowed for investigating the sensitivity of the detector's performance to its own internal parameters, and to the properties of the operating environment. The performance characteristics of Reference 1 are readily verified and quantified for environments similar to those originally described, and also for more general types of interferers, and non-Gaussian noise. In specific, for a single or small number of interferers, or in the presence of noise whose characteristics are significantly non-Gaussian, the gain afforded by the DDD is dramatic. As the noise becomes more nearly Gaussian, the gain is reduced, but in cases where analytic prediction of performance is possible, the simulation's agreement with theory is impressive.

The Vulnerability Metric's modular structure allows preprocessors such as DDD, or frequency domain based interference excision techniques, to be combined with a large range of modulation techniques, noise environments, and performance measures, to provide a thorough evaluation of communication performance and detection vulnerability.

A comparison between bit-by-bit and symbol-by-symbol decoding for a variable rate modulation scheme has been undertaken. The advantage of this scheme is that a waveform that approximates a Gaussian distribution is generated, yielding an advantage in situations where a potential interceptor can exploit the non-Gaussian distribution of typical man made spread spectrum waveforms. The systems considered held the number of generating sequences constant regardless of the desired transmitted data rate. The advantage of this approach is that the only hardware difference required as the data rate changes is in the data multiplexing circuitry. It was seen that symbol-by-symbol allows linear modulation at least up to some substantial fraction of the ultimate capacity with no asymptotic loss.

5. PROGRAM MANAGEMENT SUPPORT

The challenges for the Speakeasy program have spanned many dimensions, eg. technical, managerial, political and financial. The key to program success continues to depend on strong collaboration between the government and contractor development teams. Our major program management support focused on an initial survey of the predominant software development risks, and the implementation of proactive management techniques to control them. A summary of the progress made in resolving these issues and lessons learned is provided in this section .

5.1 RISK MANAGEMENT ASSESSMENT

One of our first tasks under this effort was to evaluate the overall Speakeasy software development process and assess the technical risks. Several areas were identified with potential critical impact on the Speakeasy program meeting its technical requirements within the current cost and schedule constraints and were singled out to receive high level management attention. These areas are identified as: contractor interaction and communications, requirements definition and traceability, and a standardized software development process. In order to realize all the program requirements within schedule and budget constraints, these concepts were translated into high priority program objectives.

- **Contractor/Team Interaction and Communications**

Creating and maintaining open and effective communication channels is a challenging problem especially when the team is geographically dispersed as in the Speakeasy program. Early in the program, much emphasis was placed on developing and fostering open communications channels between the government and contractor teams and especially in developing an effective prime /subcontractor working relationship. Frequent Technical Interchange Meetings or TIMs are held with all team members to ensure that a steady flow of information circulates throughout the project team. This is all part of a proactive management approach designed to mitigate risks through early identification, communication and correction.

- **Requirements Definition and Traceability**

Requirements must be defined at the beginning of the program to ensure that the design of the system properly incorporates all the requirements levied upon it. Like most development efforts, the Speakeasy developers are not the end-users of the final product. Therefore, understanding the requirements of the system is of paramount importance to the successful completion of the project. A common mistake in allocating system requirements is the "age-old" mentality that software is easy to change so there is no need to comprehend and analyze all the requirements before beginning to build or code the system. Inevitably, as the design progresses, requirements may be modified or new requirements may be uncovered. The Speakeasy team has devoted significant resources to developing detailed requirements up front in order to avoid costly design changes in later phases.

- **Standardized Development Process.**

The development process provides the fundamental activity guidelines for each stage of software development. Software standards should be utilized to provide program guidance within the development process. Standards that are key to the Speakeasy software development process include: DoD-STD-2167A "Defense System Software Development", and ANSI/MIL-STD-1815A "Ada Programming Language". Having project wide standards for the development process ensure that consistent procedures and tools are applied to each of the CSCIs and will facilitate their eventual integration into a total system build. DoD-STD-2167A provides guidelines for each phase of the software development process as well as standards for the form and content of any resulting products or deliverables. The Speakeasy team has put significant effort into tailoring these guidelines to meet the needs of the Speakeasy program. Having well defined and appropriate standards and procedures in place, frees the developers from unnecessary concern with the management, control and communications aspects of software development, and allows them to focus on the technical components of the CSCI.

It is important to remember that the use of standards doesn't necessarily guarantee that the deliverables will be what is expected. One must remember that standards are guidelines and that the actual content of the product can vary from one program to another. It is imperative that the team work together to tailor the standards for the individual program so that the correct amount of documentation, formal reviews, and control processes are utilized keeping in mind the specific goals of the program.

5.2 CONCLUSION

The effort required to manage a successful software development program is often misunderstood and under estimated. We believe the reasons lie with the lack of standard integrated management and development procedures and tools. During this effort, TASC has attempted to provide visibility into those areas that we have found to require close attention from day one of the Speakeasy program. The success of the Speakeasy management team has come from the realization that to meet all the am requirements within schedule and budget constraints, these concepts must be translated into priority program objectives coupled with the application of *proactive* software management techniques to the software development effort.

APPENDIX A
STATISTICS OF SUMS OF SEQUENCE

This appendix derives the second order statistics of the decision statistics necessary to evaluate the performance of the variable rate modulation scheme. Although the transmitted waveform is defined for continuous time, it is constructed to be piecewise constant, that is it changes value only at regularly spaced chip boundaries. Because of this, the waveform may be described as a vector composed of the successive values. Demodulation of the ideal staircase function is performed by integrating over successive chip intervals and manipulating the resulting discrete sequences. Observe that the noise component produced by these successive integrate-and-sample operations are independent and identically distributed.

The transmitted waveform is constructed from a collection $\{\underline{x}_i\}$ of sequences, each of length N, and with each component equiprobably set to -1 or 1, independently of all others. Suppose there are k bit streams, each using a waveform that is the sum of m of the $\{\underline{x}_i\}$. That is define

$$\underline{s}_i = \sum_{k=1}^{m} \underline{x}_{im+k} \tag{A-1}$$

If the transmitted data are defined to be b, a vector of k data bits, then the transmitted waveform is given by

$$\underline{t}(\underline{b}) = \sum_{i=1}^{k} b_i \underline{s}_i \tag{A-2}$$

The fundamental quantities necessary for evaluating error probabilities are the first and second moments of arbitrary sums of $\underline{x}_i$. If we define

$$\underline{y} = \sum_{i=1}^{L} \underline{x}_i \tag{A-3}$$

then

$$\underline{y}^T\underline{y} = \sum_{i=1}^{L}\sum_{j=1}^{L} \underline{x}_i^T\underline{x}_j \tag{A-4}$$

where $(\ )^T$ denotes transposition, and $\underline{x}^T\underline{y}$ is the conventional inner product of vectors $\underline{x}$ *and* $\underline{y}$

Since $\underline{x}_i$ *and* $\underline{x}_j$ are statistically independent for $i \neq j$, then

$$E[\underline{y}^T\underline{y}] = \sum_{i=1}^{L} E[|\underline{x}_i|^2]. \tag{A-5}$$

Each component of $\underline{x}_i$ is equal to ± 1, so $|\underline{x}_i|^2 = N$ identically. Therefore

$$E[\underline{y}^T\underline{y}] = LN. \tag{A-6}$$

We will obtain the second moment of $\underline{y}T\underline{y}$ by induction on L. Suppose

$$\underline{y} = \underline{w} + \underline{x}_L \tag{A-7}$$

where

$$\underline{w} = \sum_{i=1}^{L-1} \underline{x}_i. \tag{A-8}$$

Then

$$\underline{y}^T\underline{y} = (\underline{w} + \underline{x}_L)^T(\underline{w} + \underline{x}_L), \tag{A-9}$$

which in turn is

$$(\underline{w} + \underline{x}_L)^T(\underline{w} + \underline{x}_L) = \underline{w}^T\underline{w} + 2\underline{w}^T\underline{x}_L + \underline{x}_L^T\underline{x}_L. \tag{A-10}$$

Squaring (A-10), we obtain

(A-11)

$$(\underline{v}^T\underline{v})^2 = (\underline{w}^T\underline{w})^2 + 4(\underline{w}^T\underline{x}_L)^2 + (\underline{x}_L^T\underline{x}_L)^2 + 4(\underline{w}^T\underline{w})(\underline{w}^T\underline{x}_L) + 2(\underline{w}^T\underline{w})(\underline{x}_L^T\underline{x}_L) + 4(\underline{w}^T\underline{x}_L)(\underline{x}_L^T\underline{x}_L)$$

Taking expected values of (A-11), note that $\underline{w}$ *and* $\underline{x}_L$ are statistically independent zero mean random variables, so any term with an odd number of either quantity will have expected value zero. Performing this simplification, we obtain

$$E\left[(\underline{v}^T\underline{v})^2\right] = E\left[(\underline{w}^T\underline{w})^2\right] + 4E\left[(\underline{w}^T\underline{x}_L)^2\right] + E\left[(\underline{x}_L^T\underline{x}_L)^2\right] + 2E\left[(\underline{w}^T\underline{w})(\underline{x}_L^T\underline{x}_L)\right]. \quad \text{(A-12)}$$

The terms on the right hand side of (A-12) are taken care of as follows: the first term is the inductive quantity, that is if $E[(vTv)^2] = T_L$, *then* $E[(\underline{w}^T\underline{w})^2] = T_{L-1}$. Continuing,

$$E\left[(\underline{w}^T\underline{x}_L)^2\right] = E\left[\underline{w}^T\underline{x}_L\underline{x}_L^T\underline{w}\right]. \quad \text{(A-13)}$$

Since $\underline{w}$ *and* $\underline{x}_L$ are statistically independent,

$$E\left[(\underline{w}^T\underline{x}_L)^2\right] = E\left[\underline{w}^T E\left[\underline{x}_L\underline{x}_L^T\right]\underline{w}\right]. \quad \text{(A-14)}$$

The terms of $\underline{x}_L$ are independent and equally likely to be ± 1, so that $E[\underline{x}_L\underline{x}_L^T]$ is the identity matrix, and the second term is given by $E[\underline{w}^T\underline{w}]$, which by applying (A-6), is given by $(L-1)N$.

It was observed earlier that $\underline{x}_L^T\underline{x}_L = N$, identically, so the third term equals N^2. The last term, by applying this observation again and using (A-6), is seen to be given by $(L-1)N^2$.

Equation (A-12) is therefore given by

$$T_L = T_{L-1} + 4(L-1)N + N^2 + 2(L-1)N^2. \tag{A-15}$$

Defining

$$U_L = T_L - LN^2, \tag{A-16}$$

(A-15) becomes

$$U_L = U_{L-1} + (L-1)\,2N(N+2). \tag{A-17}$$

Observe that $T_1 = N^2$, so that $U_1 = 0$, and for $L > 1$, (A-17) can be solved to yield

$$U_L = N(N+2)\,L(L-1). \tag{A-18}$$

T_L is then given by

$$T_L = N(N+2)\,L(L-1) + LN^2. \tag{A-19}$$

In particular, then, the variance of $v^T v$ defined of course by

$$Var(\underline{v}^T\underline{v}) = E\left[(\underline{v}^T\underline{v})^2\right] - (E\left[\underline{v}^T\underline{v}\right])^2 \tag{A-20}$$

is given by

$$Var(\underline{v}^T\underline{v}) = 2NL(L-1). \tag{A-21}$$

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.